Архитектурный документ

Электронная система доставки еды из ресторанов

Дмитрий Олейник, БПИ161

1. Раздел регистрации изменений

Версия документа	Дата	Описание изменения	Автор
1.0	18.05	Создание документа	Дмитрий Олейник
1.1	23.05	Добавлены разделы 5.1.5 и 5.2.5	Дмитрий Олейник

2. Введение

2.1. Название проекта

Электронная система доставки еды из ресторанов

2.2. Рамки проекта (Scope)

В проект входит:

- система, позволяющая заказывать еду из ресторанов
- клиентское приложения для:
    - оформления заказов (пользователей),
    - доставки (курьеров)
    - администрирования (администраторов)

3. Общее описание архитектуры, задействованные архитектурные представления

Архитектура системы проектировалась по принципам SOLID, ADP (Acyclic Dependencies Principle) и CCP (Common Closure Principle), с использованием паттерна проектирования Слои (Layers).

Представление прецедентов (п. 6.1) иллюстрирует возможные сценарии использования системы, их отношения, первичные и вторичные экторов системы.

Логическое представление архитектуры (п. 6.2) иллюстрирует декомпозицию системы при помощи паттерна проектирования Слои, их роли в системе и зависимости между собой.

Представление процессов (п. 6.3) иллюстрирует процесс выполнения ключевого прецедента системы «Оформить заказ».

Представление развертывания (п. 6.4) описывает особенности развертывания системы.

Представление архитектуры безопасности (п. 6.5) описывает особенности решения проблемы безопасности системы.

Представление разработки (п. 6.6) описывает особенности и средства разработки системы.

4. Архитектурные факторы (цели и ограничения)

Заинтересованное лицо	Интерес и потребность
Архитектор ПО	Архитектура проекта должна отвечать представлениям команды разработки и проекте и отвечать поставленным требованиям (п. 5 документа-концепции).

Представитель заказчика	Продукт должен отвечать поставленным функциональным требованиям (п. 5. документа-концепции), продукт отвечать нефункциональным требованиям, особенно по части безопасности и быстродействия.

5. Технические описания архитектурных решений

5.1. Техническое описание №1

5.1.1 Проблема

Как осуществлять безопасно осуществлять оплату заказов?

5.1.2 Идея решения

Не сохранять личные данные пользователей. Использовать API банка-эквайера и осуществлять оплату через него, не сохраняя данные пользователей.

5.1.3 Факторы

При конкретной реализации система сохраняет только номера карт для удобного выбора способа оплаты пользователем.

5.1.4 Решение

Осуществление оплаты полностью производится через API банка-эквайера, которое встраивается в систему в качестве вторичного эктора.

5.1.5 Альтернативные решения

1) Разработка собственного API;
2) Разработка системы зашифрованного хранения данных.

Оба решения трудоемкие и не являются приемлемыми.

5.2. Техническое описание №2

5.2.1 Проблема

Как отложить выбор используемой базы данных на поздний срок так, чтобы это не влияло остальной разработке системы?

5.2.2 Идея решения

Использовать абстрактный слой, являющийся представлением базы данных с необходимыми функциями обращения к ней.

5.2.3 Факторы

При данном решении верхние слои архитектуры абстрагируются от конкретной базы данных, что позволяет разрабатывать их при помощи «заглушек» базы данных или использовать какую-то одну базу данных при возможности независимого перехода на другую базу данных.

При данном решении возможны эксперименты с различными базами данных в целях оценки производительности и эффективности работы системы.

5.2.4 Решение

Слой Core включает в себя два пакета: domain (представления записей в базе данных) и repository (класса для доступа к базе данных). Верхние слои получают данные через классы пакета repository в качестве объектов классов пакета domain. Верхние слои сохраняют данные через классы пакета repository, передавая в качестве аргументов объекты классов пакета domain.

Таким образом, слой Core полностью абстрагирует верхние слои (слой Service) от выбора конкретной базы данных, что позволяет отложить выбор базы данных на поздний срок и менять базу данных при изменении требований заказчика, не влияя на остальные части системы.

5.2.5 Альтернативные решения

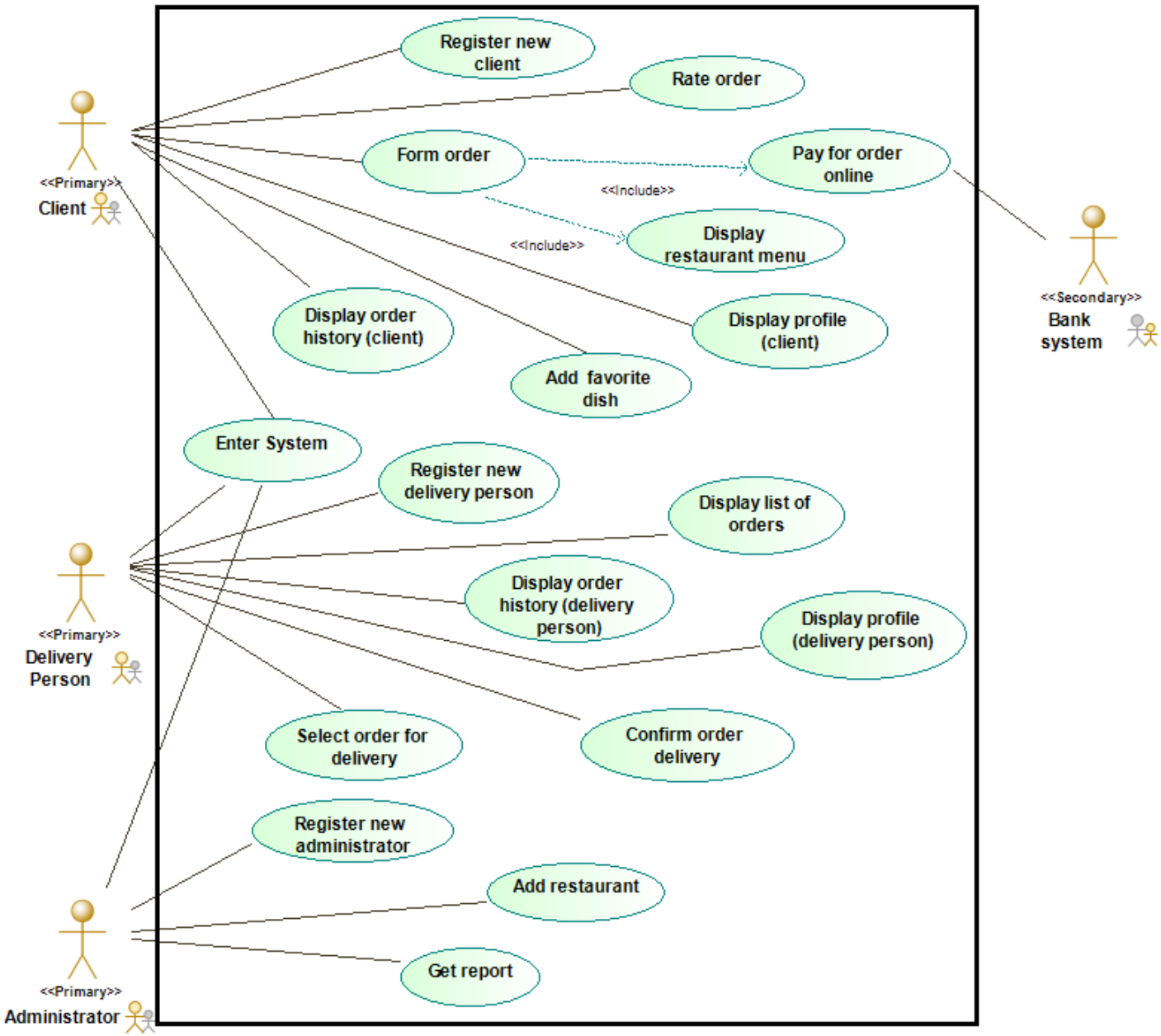
Использовать стандартные паттерны проектирования (Фасад, Фабричный метод).

Паттерн Слои выбран за счет своей наглядности не только для разработчиков, но и для бизнеса, заинтересованного в анализе архитектуры системы.

6. Представления архитектуры

В данном разделе подробно описывается каждое из используемых в проекте архитектурных представлений.

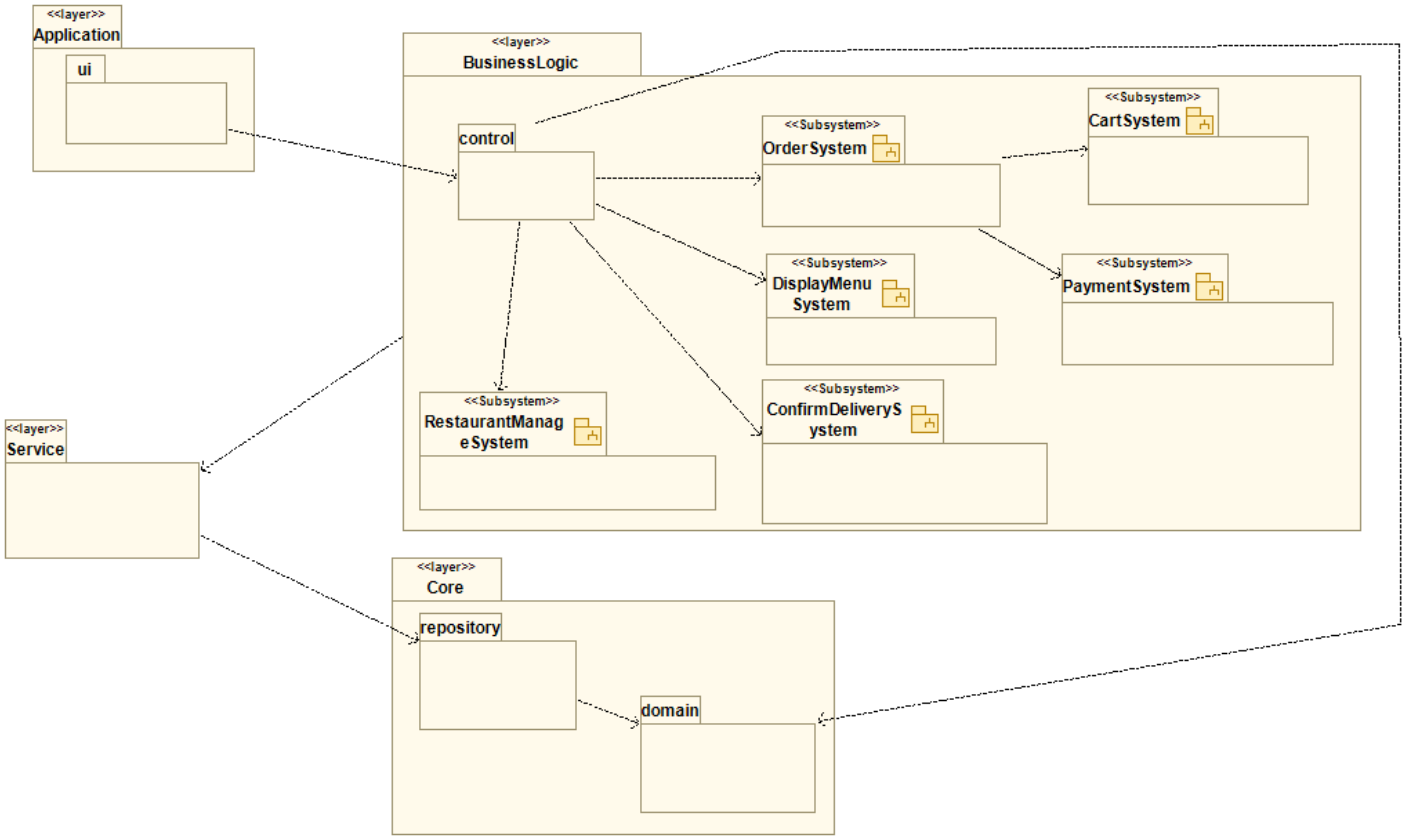
6.1. Представление прецедентов (сценариев использования)



Разработаны спецификации прецедентов «Form order», «Display restaurant menu», «Confirm order delivery», «Add restaurant» (см. папку Спецификации прецедентов).

Изображения диаграмм последовательностей для каждого прецедента см. Images/Этап3/

6.2. Логическое представление архитектуры



Система разрабатывается по слоям, каждый из которых имеет свою степень устойчивости и собственную роль в архитектуре системы:

1. *Application* – отвечает за взаимодействия с пользователем, передачи сообщений другим слоям и отображению результата работы системы.

2. *BusinessLogic* – отвечает за выполнение прецедентов и основных алгоритмов системы. Содержит основную логику системы и сильнее всего подвержен изменениям и расширениям за счет изменений требований заказчика.

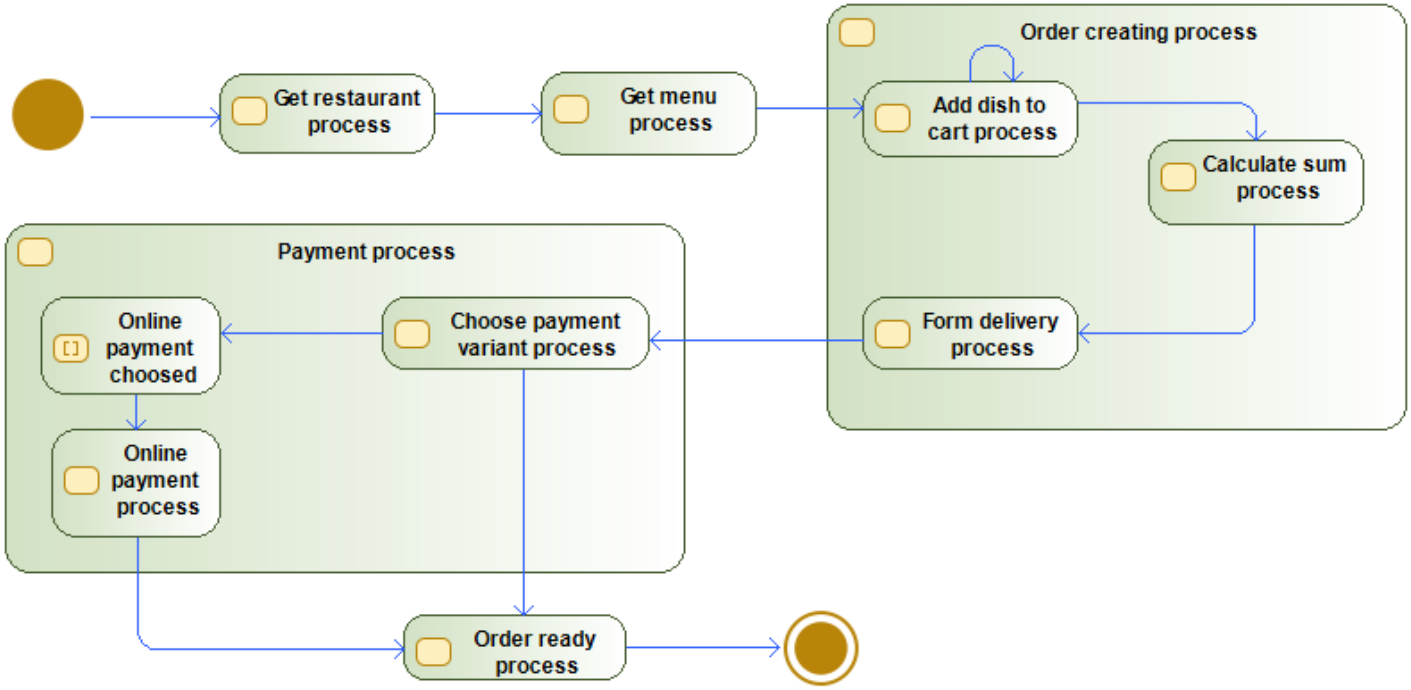
3. *Service* – отвечает за предоставление данных слою BusinessLogic в определенном виде. Включает логику различных преобразований полученных данных из слоя Core и является посредником в передаче данных между слоями BusinessLogic и Core.

4. *Core* – отвечает за взаимодействие с базой данных и представление данных в виде классов. Является абстрактным слоем, позволяющим взаимодействовать с различными типами базами данных, что позволяет оставить решение об использовании конкретной базы данных на более поздний срок.

Изображения диаграмм VOPC моделей анализа и проектирования для каждого прецедента см. Images/Этап3/ и Images/Этап4/.

6.3. Представление архитектуры процессов

Представление процесса выполнения ключевого прецедента «Оформить заказ»:



6.4. Представление развертывания

Система поставляется в виде приложения на мобильное устройство и распространяется через App Store и Google Play. Для установки, необходимо скачать приложение и магазина, соответствующего операционной системы устройства. Система также имеет интерфейс в виде веб сайта, для доступа необходимо перейти по URL.

6.5. Представление архитектуры безопасности

Для обеспечения безопасности личные данные клиента записываются в базу данных в зашифрованном виде с помощью RSA.

6.6. Представление разработки

Разработка ведется в среде разработки Intellij IDEA. Тестирование реализуется с помощью фреймворка Junit, контроль версий осуществляется с помощью Github.

6.7. Нефункциональные аспекты

- Безопасное подключение к банку-эквайеру при проведении онлайн оплаты;
- Информация о блюдах должна отражать их реальный состав (корректная конвертация введенной администратором информации);
- Безопасное хранение персональных данных клиента.

6.7.1 Объем данных и производительность системы

Время ответа на запрос не должно превышать 400 мс. Система должна выдерживать нагрузки до 3 тысяч запросов в минуту.

6.7.2 Гарантии качества работы системы

Путем нагрузочного тестирования было проверено, что сервера системы выдерживают описанную в п. 6.7.1. нагрузку.

7. Приложения

7.1. Словарь терминов

Термин	**Значение**
ПО	Программное обеспечение
Ресторан-партнёр	Предприятие общественного питания, являющееся клиентом компании «Весёлый Киви»
Курьер	Сотрудник компании «Весёлый Киви», в обязанности которого входит доставка заказов
Заказ	Набор блюд, сформированный пользователем, который должен быть доставлен по месту назначения.
Администратор	Представитель ресторана-партнёра, отвечающий за поддержку информации на сервисе в актуальном состоянии.
Курьерский «перевалочный пункт»	Логистический объект – здание компании «Весёлый Киви», предназначенное для отдыха курьеров и хранения (и аренды) курьерами корпоративных транспортных средств.